Social Coding: A Case Study with Julia

Surat Teerapittayanon*

18.337/6.338 Parallel Computing
Department of Mathematics
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology

December 18, 2011

1 Introduction

For a new programming language to success, it must pursue two goals. The first goal is a quick access to the language. The second goal is an easy reaching to other users and experts. Many new programing languages such as python and ruby address the first well by providing an online interactive prompt [4, 5]. However, they have yet addressed the second goal well enough. This project tries a new approach to address the second. It explores a new way to make it easy for user to reach to other users and experts. Specifically, this project experiments a way to create such environment for Julia [6]. We begins with a solution, a design and an implementation, followed by installation and conclusion.

2 A Solution

For Julia to achieve the second goal: an easy reaching to other users and experts, users need to be able to chat, exchange conversations and share a Julia prompt session among each other and with experts. Furthermore, they need to be able to edit the same julia program along with experts. With these features, users will be able to easily reach to other users and experts. Users can ask other users questions; in real-time, other users or experts will be able to quickly answer them, help them with their julia programs or even show them how to do it. While adding all these features, we also have to address the first goal: a quick access to the language.

To do all these, we augment the approach that has already been used by many languages (an online interactive prompt) with collaborative and social features. In fact, we augment Julia interactive prompt to allow multiple users to share a single Julia prompt session and allow them to chat among each other, may it be other users, friends or experts. Furthermore, we add a collaborative editor, allowing users to be able to edit the same Julia program. We believe that this solution addresses both goals well and it is a key toward the success of new programming languages such as Julia.

*steerapi@mit.edu

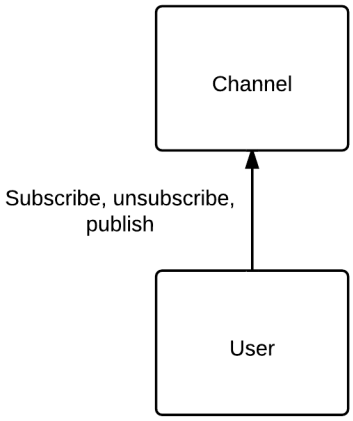

Figure 1: An object model the basis of the application.

3 A Design

To achieve both goals, we decide to create a real-time single-page web application [1]. The basis of the application is the object model depicted in Figure 1. Each user subscribes to (1) a channel identified with his or her own name and (2) the public channel. The design mockup of the application is shown in Figure 2. Three main features include chat with other users, shared julia prompt session and a collaborative editor. First, when a user clicked on a name on the right panel, a chat box will pop up and the user will be able to type and chat with the other user. A chat message will be published to the channel identified with the recipient's name. Second, users can type in Julia command in the prompt. The command and its respond will be published to the public channel; thus, they will be received by all users. Third, when users clicked on the "Editor" button, they will switch from Prompt screen to Editor screen. In this screen, user will be able to collaboratively edit a Julia program. In addition, they can run this program in Prompt screen by clicking "Run Code" button.

4 An Implementation

In this section, we describe an implementation of the design. We use two main open-source projects: Socketstream [2] and Etherpad-lite [3]. We use Socketstream for chat and shared Julia prompt session; we use Etherpad-lite for collaborative editing.

4.1 Chat and Shared Julia Prompt Session: Socketstream

To implement chat and shared Julia prompt session, we use Socketstream framework. Socketstream provides the following features:

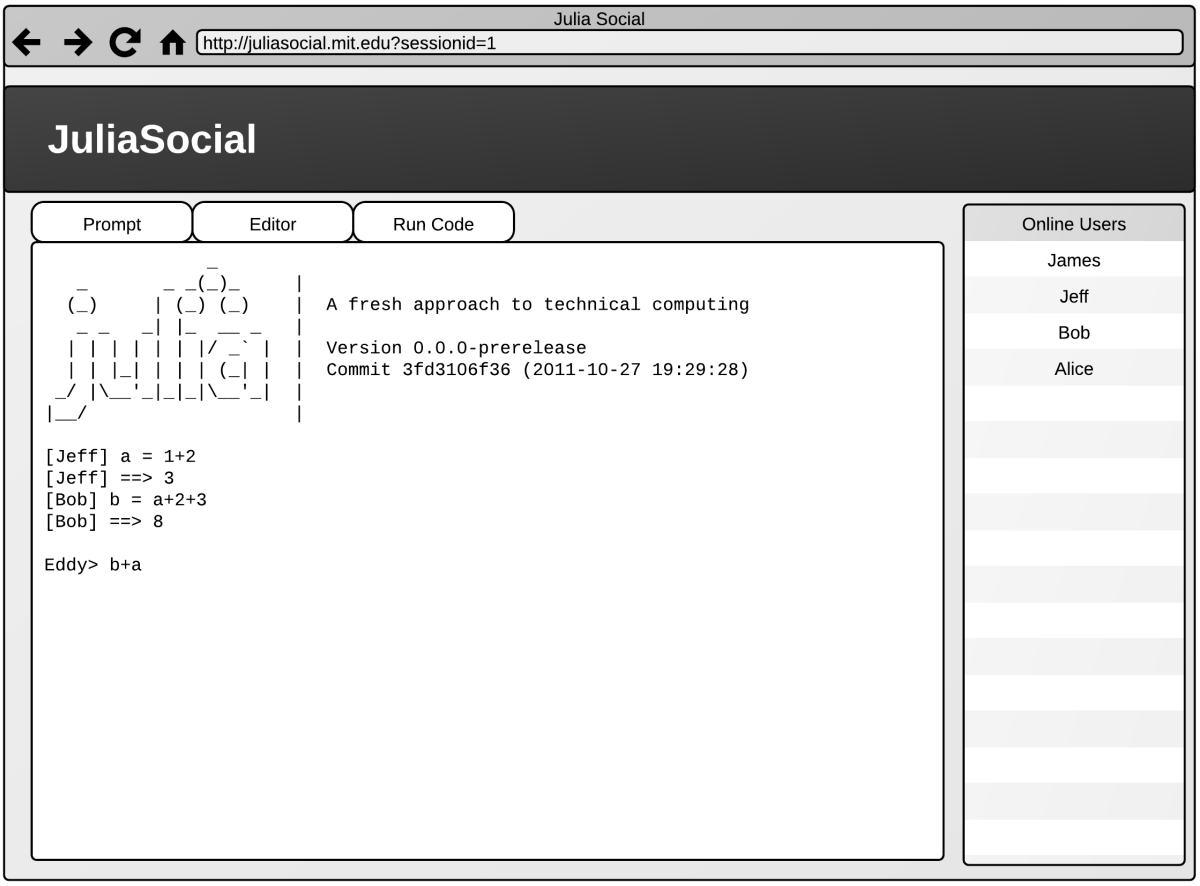

Figure 2: A design mockup of the application.

- NodeJS: a server-side javascript [**?**].
- Coffeescript support [12].
- Jade Template Engine support [13].
- Socket.IO support [14].
- ZeroMQ support [16].
- Redis support [15].

For more information about Socketstream, please visit:
http://github.com/socketstream/socketstream.

4.2 Collaborative Editing: Etherpad-lite

Etherpad-lite is an online collaborative editing. We make use of Etherpad-lite by embedding it into our web application via iframe; we get contents of the editor using Etherpad-lite jQuery Plugin [11].

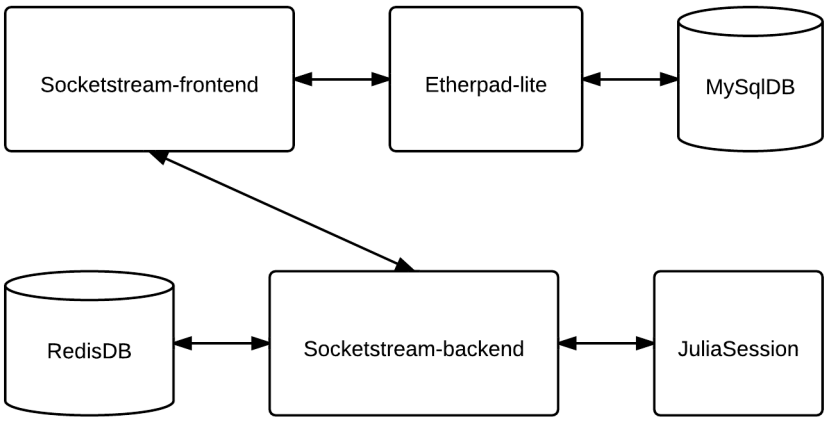

Figure 3: An application architecture.

For more information about Etherpad-lite, please visit:
https://github.com/Pita/etherpad-lite.

4.3 Libraries

Libraries used in this implementation are:

- jQuery Plugin. http://jquery.com/.
- jQuery UI. http://jqueryui.com/
- jQuery UI Chatbox. http://www.cs.illinois.edu/homes/wenpu1/chatbox.html.
- jQuery Purr. http://code.google.com/p/jquery-purr/.
- Etherpad-lite jQuery Plugin. https://github.com/johnyma22/etherpad-lite-jquery-plugin.
- Mustache. http://mustache.github.com/.

Putting Socketstream and Etherpad-lite together, we have an architecture for our application (Figure 3). This architecture allows running a Julia program in Editor screen in Prompt screen. When a user clicks "Run Code" button, Socketstream-frontend gets contents of the editor from Etherpad-lite (usually stored inside MysqlDB) and passes the contents to Socketstream-backend. Socketstream-backend executes the contents in a JuliaSession and publishes to a public channel.

Snapshots of the application are illustrated in Figure 5 and Figure 4.

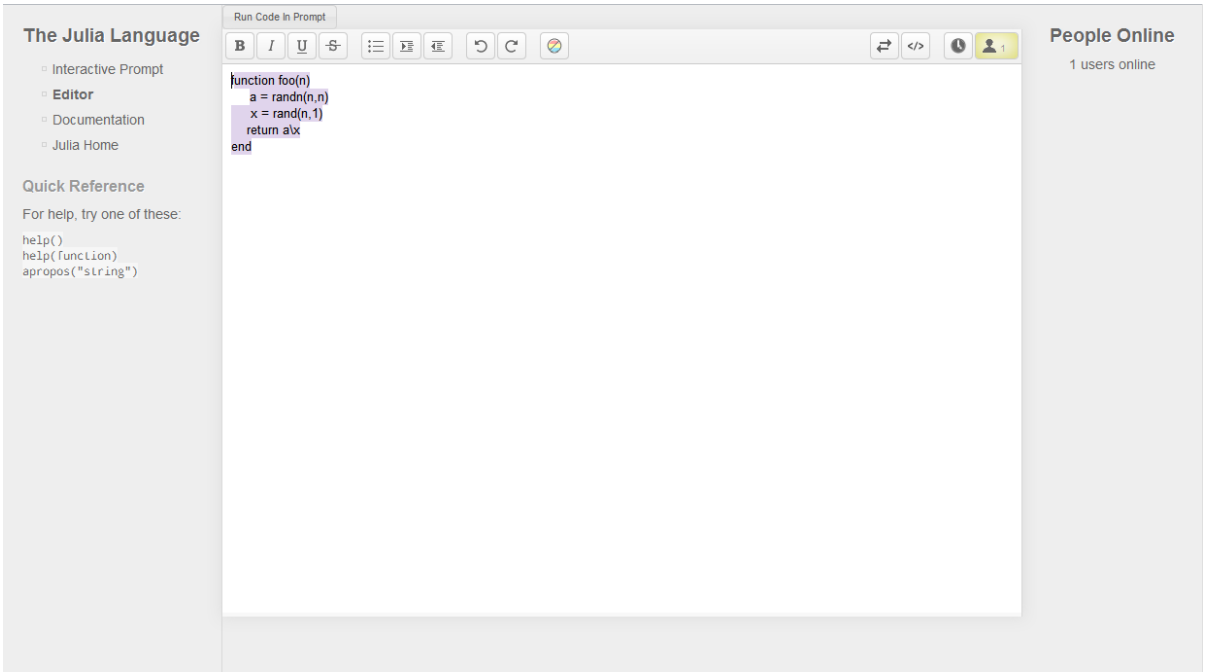

Figure 4: A snapshot of the application's editor screen.

5 Installation

This section describes what you need to install this application. The application requires Redis, NodeJS, Etherpad-lite, Socketstream and Julia.

5.1 Redis Installation

Please follow the instruction at http://redis.io/download.

5.2 NodeJS Installation

Please follow the instruction at http://nodejs.org/.

5.3 Etherpad-lite Installation

Please follow the instruction at https://github.com/Pita/etherpad-lite.

5.4 Socketstream Installation

Please follow the instruction at http://github.com/socketstream/socketstream or execute:

```
npm install -g socketstream
```

5.5 Julia Installation

Please follow the instruction at https://github.com/JuliaLang/julia

Figure 5: A snapshot of the application's prompt screen.

5.6 Application Installation

First of all, please install git if you have yet had git. You can get git here: http://git-scm.com/. To install the application:

```
git clone git://github.com/Wisdom/juliaSocial.git
cd juliaSocial
npm install node-uuid
```

5.7 Application Configuration

The configuration file is inside config/app.coffee. Please point "julia" to your julia binary and "etherpad" to your etherpad host.

5.8 Running the application

First, start redis server:

```
redis-server
```

Second, inside the juliaSocial folder run:

```
socketstream start
```

If everything goes well, your application should be running. Visit http://localhost:3000 to see the application in action.

For the latest source codes and updates, please visit:
http://github.com/Wisdom/juliaSocial.

6 Conclusion

In this project, we provide a solution, a design and an implementation to the two goals needed for the success of a new programming language. Further development of this project will continue at http://github.com/Wisdom/juliaSocial. Future work includes Julia syntax parser, multiple sessions and rooms, multiple programming languages, Facebook and Twitter integration and Julia syntax highlighter. We hope that this project will accelerate the adoption of Julia; we also hope that it will be useful to other programming languages as well.

Finally, I would like to thank Professor Alan Edelman, Jeff Bezanson, Stephan Boyer, developers of all the projects and libraries used in this project, Julia developers and the class.

References

[1] Single Page Application. http://en.wikipedia.org/wiki/Single-page_application.

[2] Socketstream. http://github.com/socketstream/socketstream.

[3] Etherpad-lite. https://github.com/Pita/etherpad-lite.

[4] Try Python. http://trypython.org/.

[5] Try Ruby. http://tryruby.org/.

[6] Julia. https://github.com/JuliaLang/julia.

[7] jQuery Console. https://github.com/replit/jq-console.

[8] jQuery UI. http://jqueryui.com/.

[9] jQuery UI Chatbox. https://github.com/dexterpu/jquery.ui.chatbox.

[10] jQuery Purr. http://code.google.com/p/jquery-purr/.

[11] Etherpad-lite jQuery Plugin. https://github.com/johnyma22/etherpad-lite-jquery-plugin.

[12] NodeJS. http://nodejs.org/.

[13] Coffeescript. http://coffeescript.org/.

[14] Jade. http://jade-lang.com/.

[15] Socket.IO. http://socket.io/.

[16] Redis. http://redis.io/.

[17] ZeroMQ. http://www.zeromq.org/.